# 作业

1. GB2312

来历：《信息交换用汉字编码字符集》是由中国国家标准总局1980年发布，1981年5月1日开始实施的一套国家标准，标准号是GB2312。

产生原因：当中国人们得到计算机时，已经没有可以利用的字节状态来表示汉字，况且有6000多个常用汉字需要保存，于是想到把那些ASCII码中127号之后的奇异符号们直接取消掉, 规定：一个小于127的字符的意义与原来相同，但两个大于127的字符连在一起时，就表示一个汉字，前面的一个字节（称之为高字节）从0xA1用到0xF7，后面一个字节（低字节）从0xA1到0xFE，这样我们就可以组合出大约7000多个简体汉字了。在这些编码里，我们还把数学符号、罗马希腊的字母、日文的假名们都编进去了，连在 ASCII 里本来就有的数字、标点、字母都统统重新编了两个字节长的编码，这就是常说的"全角"字符，而原来在127号以下的那些就叫"半角"字符了。这种汉字方案叫做 "GB2312"。GB2312 是对 ASCII 的中文扩展。

解决的问题：汉字的存储问题。

字节数：2

描述：国标码，专门用来表示汉字，是双字节编码，而英文字母和iso8859-1一致（兼容iso8859-1编码）。其中gbk编码能够用来同时表示繁体字和简体字，而gb2312只能表示简体字

兼容ASCII。

相互的转换：

2.big5

来历：‘大五码’（Big5）是由台湾[财团法人信息产业策进会](https://zh.wikipedia.org/wiki/%E8%B3%87%E8%A8%8A%E5%B7%A5%E6%A5%AD%E7%AD%96%E9%80%B2%E6%9C%83" \o "信息产业策进会)为[五大中文套装软件](https://zh.wikipedia.org/wiki/%E4%BA%94%E5%A4%A7%E4%B8%AD%E6%96%87%E5%A5%97%E8%A3%9D%E8%BB%9F%E9%AB%94" \o "五大中文套装软件)所设计的中文共通内码，在1983年12月完成公告，隔年3月，信息产业策进会与[台湾](https://zh.wikipedia.org/wiki/%E8%87%BA%E7%81%A3" \o "台湾)13家厂商签定“16位个人电脑套装软件合作开发（BIG-5）项目（[五大中文套装软件](https://zh.wikipedia.org/wiki/%E4%BA%94%E5%A4%A7%E4%B8%AD%E6%96%87%E5%A5%97%E8%A3%9D%E8%BB%9F%E9%AB%94" \o "五大中文套装软件)）”，因为此中文内码是为台湾自行制作开发之“五大中文套装软件”所设计的，所以就称为Big5中文内码。五大中文套装软件虽然并没有如预期的取代国外的套装软件，但随着采用Big5码的[国乔中文系统](https://zh.wikipedia.org/w/index.php?title=%E5%9C%8B%E5%96%AC%E4%B8%AD%E6%96%87%E7%B3%BB%E7%B5%B1&action=edit&redlink=1" \o "国乔中文系统（页面不存在）)及[倚天中文系统](https://zh.wikipedia.org/wiki/%E5%80%9A%E5%A4%A9%E4%B8%AD%E6%96%87%E7%B3%BB%E7%B5%B1" \o "倚天中文系统)先后在台湾市场获得成功，使得Big5码深远地影响繁体中文电脑[内码](https://zh.wikipedia.org/wiki/%E5%85%A7%E7%A2%BC" \o "内码)，直至今日。“五大码”的英文名称“Big5”后来被人按英文字序译回中文，以致现在有“五大码”和“大五码”两个中文名称。

解决的问题：Big5码的产生，是因为当时[个人电脑](https://zh.wikipedia.org/wiki/%E5%80%8B%E4%BA%BA%E9%9B%BB%E8%85%A6" \o "个人电脑)没有共通的内码，导致厂商推出的中文应用软件无法推广，并且与[IBM 5550](https://zh.wikipedia.org/wiki/IBM_5550" \o "IBM 5550)、[王安码](https://zh.wikipedia.org/w/index.php?title=%E7%8E%8B%E5%AE%89%E7%A2%BC&action=edit&redlink=1" \o "王安码（页面不存在）)等内码，彼此不能兼容；另一方面，台湾当时尚未推出中文编码标准。在这样的时空背景下，为了使台湾早日进入信息时代，所采行的一个项目；同时，这个项目对于以台湾为核心的亚洲繁体汉字圈也产生了久远的影响。

字节数：Big5码是一套[双字节字符集](https://zh.wikipedia.org/wiki/%E5%8F%8C%E5%AD%97%E8%8A%82%E5%AD%97%E7%AC%A6%E9%9B%86" \o "双字节字符集)，使用了双八码存储方法，以两个字节来安放一个字。第一个字节称为“高位字节”，第二个字节称为“低位字节”。“高位字节”使用了0x81-0xFE，“低位字节”使用了0x40-0x7E，及0xA1-0xFE。

描述：Big5，又称为大五码或五大码，是使用[繁体中文](https://zh.wikipedia.org/wiki/%E7%B9%81%E4%BD%93%E4%B8%AD%E6%96%87" \o "繁体中文)（正体中文）社区中最常用的电脑[汉字](https://zh.wikipedia.org/wiki/%E6%B1%89%E5%AD%97" \o "汉字)[字符集](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E9%9B%86" \o "字符集)标准，共收录13,060个汉字

1. Unicode

来历：Unicode伴随着[通用字符集](https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E5%AD%97%E7%AC%A6%E9%9B%86" \o "通用字符集)的标准而发展，同时也以书本的形式对外发表。Unicode至今仍在不断增修，每个新版本都加入更多新的字符。目前最新的版本为2017年6月20日公布的10.0.0，已经收录超过十万个[字符](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6" \o "字符)（第十万个字符在2005年获采纳）。Unicode涵盖的数据除了视觉上的字形、编码方法、标准的[字符编码](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E7%BC%96%E7%A0%81" \o "字符编码)外，还包含了字符特性，如大小写字母。

产生原因：Unicode是为了解决传统的[字符编码](https://zh.wikipedia.org/wiki/%E5%AD%97%E5%85%83%E7%B7%A8%E7%A2%BC" \o "字符编码)方案的局限而产生的，例如[ISO 8859-1](https://zh.wikipedia.org/wiki/ISO/IEC_8859" \o "ISO/IEC 8859)所定义的字符虽然在不同的国家中广泛地使用，可是在不同国家间却经常出现不兼容的情况。很多传统的编码方式都有一个共同的问题，即容许电脑处理双语环境（通常使用[拉丁字母](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E5%AD%97%E6%AF%8D" \o "拉丁字母)以及其本地语言），但却无法同时支持多语言环境（指可同时处理多种语言混合的情况）。

解决的问题：

字节数：目前实际应用的统一码版本对应于[UCS-2](https://zh.wikipedia.org/wiki/UCS-2" \o "UCS-2)，使用16[位](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83" \o "比特)的编码空间。也就是每个字符占用2个[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82" \o "字节)。这样理论上一共最多可以表示216（即65536）个字符。基本满足各种语言的使用。实际上当前版本的统一码并未完全使用这16位编码，而是保留了大量空间以作为特殊使用或将来扩展。

上述16位统一码字符构成[基本多文种平面](https://zh.wikipedia.org/wiki/%E5%9F%BA%E6%9C%AC%E5%A4%9A%E6%96%87%E7%A8%AE%E5%B9%B3%E9%9D%A2" \o "基本多文种平面)。最新（但未实际广泛使用）的统一码版本定义了16个[辅助平面](https://zh.wikipedia.org/wiki/%E8%BE%85%E5%8A%A9%E5%B9%B3%E9%9D%A2" \o "辅助平面)，两者合起来至少需要占据21位的编码空间，比3字节略少。但事实上辅助平面字符仍然占用4字节编码空间，与[UCS-4](https://zh.wikipedia.org/wiki/UCS-4" \o "UCS-4)保持一致。未来版本会扩充到ISO 10646-1实现级别3，即涵盖UCS-4的所有字符。UCS-4是一个更大的尚未填充完全的31位字符集，加上恒为0的首位，共需占据32位，即4字节。理论上最多能表示231个字符，完全可以涵盖一切语言所用的符号。

兼容Unicode字符集有多个编码方式，分别是UTF-8，UTF-16和UTF-32。但不兼容GB2312。

相互的转换：

1. Utf-8

来历：1992年初，为创建良好的[字节串编码系统](https://zh.wikipedia.org/w/index.php?title=%E4%BD%8D%E5%85%83%E7%B5%84%E4%B8%B2%E7%B7%A8%E7%A2%BC%E7%B3%BB%E7%B5%B1&action=edit&redlink=1" \o "字节串编码系统（页面不存在）)以供多[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82" \o "字节)字符集使用，开始了一个正式的研究。[ISO/IEC 10646](https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E5%AD%97%E7%AC%A6%E9%9B%86" \o "通用字符集)的初稿中有一个非必须的附录，名为UTF。当中包含了一个供32[比特](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83" \o "比特)的[字符](https://zh.wikipedia.org/wiki/UCS" \o "UCS)使用的字节串编码系统。这个编码方式的性能并不令人满意，但它提出了将0-127的范围保留给ASCII以兼容旧系统的概念。

1992年7月，[X/Open](https://zh.wikipedia.org/wiki/X/Open" \o "X/Open)委员会XoJIG开始寻求一个较佳的编码系统。[Unix系统实验室](https://zh.wikipedia.org/wiki/Unix%E7%B3%BB%E7%BB%9F%E5%AE%9E%E9%AA%8C%E5%AE%A4" \o "Unix系统实验室)（USL）的Dave Prosser为此提出了一个编码系统的建议。它具备可更快速实现的特性，并引入一项新的改进。其中，7[比特](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83" \o "比特)的[ASCII](https://zh.wikipedia.org/wiki/ASCII" \o "ASCII)符号只代表原来的意思，所有多字节序列则会包含第8[比特](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83" \o "比特)的符号，也就是所谓的[最高有效比特](https://zh.wikipedia.org/wiki/%E6%9C%80%E9%AB%98%E6%9C%89%E6%95%88%E4%BD%8D%E5%85%83" \o "最高有效比特)。

1992年8月，这个建议由[IBM](https://zh.wikipedia.org/wiki/IBM" \o "IBM)[X/Open](https://zh.wikipedia.org/wiki/X/Open" \o "X/Open)的代表流传到一些感兴趣的团体。与此同时，[贝尔实验室](https://zh.wikipedia.org/wiki/%E8%B2%9D%E7%88%BE%E5%AF%A6%E9%A9%97%E5%AE%A4" \o "贝尔实验室)[九号项目](https://zh.wikipedia.org/wiki/%E8%B2%9D%E7%88%BE%E5%AF%A6%E9%A9%97%E5%AE%A4%E4%B9%9D%E8%99%9F%E8%A8%88%E7%95%AB" \o "贝尔实验室九号项目)[操作系统](https://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1" \o "操作系统)工作小组的[肯·汤普逊](https://zh.wikipedia.org/wiki/%E8%82%AF%C2%B7%E6%B1%A4%E6%99%AE%E9%80%8A" \o "肯·汤普逊)对这编码系统作出重大的修改，让编码可以自我同步，使得不必从字符串的开首读取，也能找出字符间的分界。1992年9月2日，[肯·汤普逊](https://zh.wikipedia.org/wiki/%E8%82%AF%C2%B7%E6%B1%A4%E6%99%AE%E9%80%8A" \o "肯·汤普逊)和[罗勃·派克](https://zh.wikipedia.org/wiki/%E7%BE%85%E5%8B%83%C2%B7%E6%B4%BE%E5%85%8B" \o "罗勃·派克)一起在[美国](https://zh.wikipedia.org/wiki/%E7%BE%8E%E5%9C%8B" \o "美国)[新泽西州](https://zh.wikipedia.org/wiki/%E6%96%B0%E6%BE%A4%E8%A5%BF%E5%B7%9E" \o "新泽西州)一架餐车的餐桌垫上描绘出此设计的要点。接下来的日子，Pike及汤普逊将它实现，并将这编码系统完全应用在[九号项目](https://zh.wikipedia.org/wiki/%E8%B2%9D%E7%88%BE%E5%AF%A6%E9%A9%97%E5%AE%A4%E4%B9%9D%E8%99%9F%E8%A8%88%E7%95%AB" \o "贝尔实验室九号项目)当中，及后他将有关成果回馈X/Open。

1993年1月25-29日的在[圣地牙哥](https://zh.wikipedia.org/wiki/%E8%81%96%E5%9C%B0%E7%89%99%E5%93%A5" \o "圣地牙哥)举行的[USENIX](https://zh.wikipedia.org/wiki/USENIX" \o "USENIX)会议首次正式介绍UTF-8。

自1996年起，[微软](https://zh.wikipedia.org/wiki/%E5%BE%AE%E8%BB%9F" \o "微软)的[CAB](https://zh.wikipedia.org/wiki/CAB" \o "CAB)（MS Cabinet）规格在UTF-8标准正式落实前就明确容许在任何地方使用UTF-8编码系统。但有关的编码器实际上从来没有实现这方面的规格。

产生原因：

解决的问题：事实证明，对可以用[ASCII](https://baike.baidu.com/item/ASCII" \t "_blank)表示的[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6" \t "_blank)使用UNICODE并不高效，因为UNICODE比ASCII占用大一倍的空间，而对ASCII来说高[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82" \t "_blank)的0对他毫无用处。为了解决这个问题，就出现了一些中间格式的[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86" \t "_blank)，他们被称为通用转换格式，即[UTF](https://baike.baidu.com/item/UTF" \t "_blank)（Unicode Transformation Format）。常见的UTF格式有：UTF-7, UTF-7.5, UTF-8,[UTF-16](https://baike.baidu.com/item/UTF-16" \t "_blank), 以及 [UTF-32](https://baike.baidu.com/item/UTF-32" \t "_blank)。

字节数：UTF-8使用一至六个[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82" \o "字节)为每个字符编码（尽管如此，2003年11月UTF-8被RFC 3629重新规范，只能使用原来Unicode定义的区域，U+0000到U+10FFFF，也就是说最多四个字节）：

1. 128个US-ASCII字符只需一个字节编码（Unicode范围由U+0000至U+007F）。
2. 带有[附加符号](https://zh.wikipedia.org/wiki/%E9%99%84%E5%8A%A0%E7%AC%A6%E5%8F%B7" \o "附加符号)的[拉丁文](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E6%96%87" \o "拉丁文)、[希腊文](https://zh.wikipedia.org/wiki/%E5%B8%8C%E8%87%98%E6%96%87" \o "希腊文)、[西里尔字母](https://zh.wikipedia.org/wiki/%E8%A5%BF%E9%87%8C%E7%88%BE%E5%AD%97%E6%AF%8D" \o "西里尔字母)、[亚美尼亚语](https://zh.wikipedia.org/wiki/%E4%BA%9E%E7%BE%8E%E5%B0%BC%E4%BA%9E%E8%AA%9E" \o "亚美尼亚语)、[希伯来文](https://zh.wikipedia.org/wiki/%E5%B8%8C%E4%BC%AF%E4%BE%86%E6%96%87" \o "希伯来文)、[阿拉伯文](https://zh.wikipedia.org/wiki/%E9%98%BF%E6%8B%89%E4%BC%AF%E6%96%87" \o "阿拉伯文)、[叙利亚文](https://zh.wikipedia.org/wiki/%E5%8F%99%E5%88%A9%E4%BA%9A%E6%96%87" \o "叙利亚文)及[它拿字母](https://zh.wikipedia.org/wiki/%E5%AE%83%E6%8B%BF%E5%AD%97%E6%AF%8D" \o "它拿字母)则需要两个字节编码（Unicode范围由U+0080至U+07FF）。
3. 其他[基本多文种平面](https://zh.wikipedia.org/wiki/%E5%9F%BA%E6%9C%AC%E5%A4%9A%E6%96%87%E7%A8%AE%E5%B9%B3%E9%9D%A2" \o "基本多文种平面)（BMP）中的字符（这包含了大部分常用字，如大部分的汉字）使用三个字节编码（Unicode范围由U+0800至U+FFFF）。
4. 其他极少使用的Unicode [辅助平面](https://zh.wikipedia.org/wiki/%E8%BC%94%E5%8A%A9%E5%B9%B3%E9%9D%A2" \o "辅助平面)的字符使用四至六字节编码（Unicode范围由U+10000至U+1FFFFF使用四字节，Unicode范围由U+200000至U+3FFFFFF使用五字节，Unicode范围由U+4000000至U+7FFFFFFF使用六字节）。

相互转换：在UTF-8 编码中，所有的字符均使用1 到6 个字节进行

编码。在只包含1 个字节的UTF-8 编码中，其最高位置0，其余

的7 个二进制位用来对字符进行编码；在含n （ 1<n!6）个字节

的UTF-8 编码中，其第一个字节的前n 位置1，第n+1 位置0，

该字节剩余的其它二进制位用来对字符进行编码，后续字节的

最高位均置1，次高位均置0，每个字节中剩余的6 个二进制用

来对字符进行编码。一个字符的UTF-8 编码中包含字节个数

的多少是由其Unicode 编码所处的范围来决定的，具体如表1

所示，该表中给出了不同范围内的Unicode 字符对应的UTF-8

编码的格式，其中已填入的1、0 数据表示约定的标记位，x 表示

对字符编码时可用的二进制位。将Unicode 编码的二进制位按

由低到高的次序放入x 表示的空位中即可得到其所对应的UTF-8 编码。

Utf-16:

描述：Unicode的编码空间从U+0000到U+10FFFF，共有1,112,064个码位（code point）可用来映射字符. Unicode的编码空间可以划分为17个平面（plane），每个平面包含216（65,536）个码位。17个平面的码位可表示为从U+xx0000到U+xxFFFF，其中xx表示十六进制值从0016到1016，共计17个平面。第一个平面称为基本多语言平面（Basic Multilingual Plane, BMP），或称第零平面（Plane 0）。其他平面称为辅助平面（Supplementary Planes）。基本多语言平面内，从U+D800到U+DFFF之间的码位区块是永久保留不映射到Unicode字符。UTF-16就利用保留下来的0xD800-0xDFFF区段的码位来对辅助平面的字符的码位进行编码。

字节数：UTF-16是[Unicode](https://zh.wikipedia.org/wiki/Unicode" \o "Unicode)字符编码五层次模型的第三层：字符编码表（Character Encoding Form，也称为"storage format"）的一种实现方式。即把Unicode字符集的抽象[码位](https://zh.wikipedia.org/wiki/%E7%A0%81%E4%BD%8D" \o "码位)映射为16位长的整数（即[码元](https://zh.wikipedia.org/wiki/%E7%A0%81%E5%85%83" \o "码元)）的序列，用于数据存储或传递。Unicode字符的码位，需要1个或者2个16位长的码元来表示，因此这是一个变长表示。

相互的转换：UTF-16比起[UTF-8](https://zh.wikipedia.org/wiki/UTF-8" \o "UTF-8)，好处在于大部分字符都以固定长度的字节（2字节）存储，但UTF-16却无法兼容于[ASCII](https://zh.wikipedia.org/wiki/ASCII" \o "ASCII)编码。

UTF-16的大尾序和小尾序存储形式都在用。一般来说，以[Macintosh](https://zh.wikipedia.org/wiki/Macintosh" \o "Macintosh)制作或存储的文字使用大尾序格式，以[Microsoft](https://zh.wikipedia.org/wiki/Microsoft" \o "Microsoft)或[Linux](https://zh.wikipedia.org/wiki/Linux" \o "Linux)制作或存储的文字使用小尾序格式。